UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P O Box 1450
Alexandria, Virginia 22313-1450
www uspto gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/830,042	04/23/2004	John M. Holt	32199-0384097/5027B-US	9699

93729 7590 04/15/2010
Pillsbury Winthrop Shaw Pittman LLP
Waratek
P.O. Box 10500
McLean, VA 22102

EXAMINER	
RUTTEN, JAMES D	

ART UNIT	PAPER NUMBER
2192	

NOTIFICATION DATE	DELIVERY MODE
04/15/2010	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

docket_ip@pillsburylaw.com

Office Action Summary	**Application No.** 10/830,042	**Applicant(s)** HOLT, JOHN M.
	Examiner JAMES RUTTEN	**Art Unit** 2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *18 December 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *7-9,15-18,24-26 and 29-53* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *7-9,15-18,24-26 and 29-51* is/are rejected.

7)☒ Claim(s) *52 and 53* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *12/23/09*.	6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to Applicant's submission filed 12/18/09, responding to the 8/26/09 Office action which detailed the rejection of claims 7-9, 15-18, 24-26, and 29-47. Claims 7, 9, 15-18, 26 have been amended, claim 47 has been canceled, and new claims 48-53 have been added. Claims 7-9, 15-18, 24-26, and 29-53 remain pending in the application and have been fully considered by the examiner.

Information Disclosure Statement

2. Citation "CC" of the 12/23/09 IDS provides the following: "Office Action received in co-pending application number 11/111,946; Attorney Docket Number 1130-8110.US06) (5027F-US}mailed on September 26, 2006." However, no Office Action mailed 9/26/06 was found in 11/111,946. An Office action mailed 9/26/08 was found and has been considered.

Response to Amendment/Arguments

3. The 12/18/09 amendments have overcome the prior claim objections, which have been accordingly withdrawn.

4. Applicant's arguments, see pages 15-17, filed 12/18/2009, with respect to 35 U.S.C. § 112, first paragraph, have been fully considered and are persuasive. The rejection of claims 33, 41, and 44 has been withdrawn.

5. The 12/18/09 amendments have overcome the prior claim rejections under 35 U.S.C. § 112, second paragraph, which have been accordingly withdrawn.

6. On page 19 filed 12/18/09, Applicant essentially argues that prior art of record Factor describes a distributed shared memory arrangement where each computer is able to read from the memory of other computers, and therefore the total memory available to each computer is the sum of the memory of all computers. Applicant supports the argument by citing Factor Part 1 page 2 (left-hand column), and Part 3 which describe "distributed shared memory (DSM)." Applicant further argues that Factor's distributed shared memory cannot teach the claimed "independent local memory." However, review of section 3 describing Factor's distributed shared memory uses a protocol called *Multithreaded Scalable Home-based Lazy Release Consistency*. In the fourth paragraph of section 3, Factor describes the notion of a *home-based* protocol which detects updates to objects which are then propagated. All copies of an object are cached from the master copy maintained at the home. As such, Factor's memory model does not allow direct reading of the memory of another computer. Nonetheless, Factor does not clearly describe the process of acquiring object updates. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection is made in view of "Concurrency Control and View Notification Algorithms for Collaborative Replicated Objects" by Strom et al.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claim 7, 8, 9, 24, 25, 29, 30, 33, 48, 49, and 51 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

9. The term "substantially" in claim 7 is a relative term which renders the claim indefinite. The term "substantially" is not defined by the claim, the specification does not provide a standard for ascertaining the requisite degree, and one of ordinary skill in the art would not be reasonably apprised of the scope of the invention. The claimed term "simultaneously" is rendered indefinite by using the relative term "substantially." Claims 49 and 51 use the term "substantially" and are rejected for the same reason. Claims 8, 9, 24, 25, 29, 30, 48, and 49 are dependent upon claim 7 and are rejected as being dependent upon a rejected base claim.

10. Claim 33 recites the limitation "the alerted thread" in line 2. There is insufficient antecedent basis for this limitation in the claim. As noted above, this limitation has issues regarding the written description requirement. For the purpose of further examination, this limitation will be interpreted as "*[code thread that alerts the DRT]*" as supported on page 7 lines 19-23.

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. Claims 7, 8, and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over "JavaSplit: A Runtime for Execution of Monolithic Java Programs on Heterogeneous Collections of CommodityWorkstations" by Factor et al. ("Factor") in view of "Concurrency Control and View Notification Algorithms for Collaborative Replicated Objects" by Strom et al. (hereinafter "Strom").

In regard to claim 7, Factor discloses:

In a multiple computer system including a plurality of single computers interconnected via a communications link, a method of loading an application program onto each of said plurality of single computers, the application program having application program code including a plurality of code threads all intended to execute on and reference a single computer having a single processing unit or symmetric multiple processing units and a single independent local memory with a local memory capacity that is not shared with any other single computer of said plurality of single computers, See at least page 1 right column, e.g.: "It is able to execute a pre-existing, multithreaded application on any given heterogeneous set of machines." Note that the system described in Factor utilizes the Java programming environment which is understood to execute first and foremost on a single processing unit with a single independent local memory. Factor's disclosure implies this by suggesting that their system can take such an application and execute it in a multiple computer system.

the method comprising the steps of:

loading the application program written to operate only on a single computer onto each different computer of said plurality of single computers; and See the top of the left column on page 2: "In JavaSplilt, each node executes its part of the rewritten application on its local standard JVM."

modifying the application program on each said different computer before execution of said corresponding portion of the application program written to operate only on a single computer on each said different single computer; See section 4 on page 3, e.g. "Bytecode instrumentation."

substantially simultaneously executing different portions of said application program on each different one of the plurality of single computers with each different one of the plurality of single computers having a different independent local memory accessible only by a corresponding portion of the application program. See page 1, right column, e.g.: "When combined with the code of the runtime, the instrumented program becomes a distributed application, ready to be executed on multiple nodes (Figure 1)."

Factor discloses providing cached copies of an object for further execution on each node. See section 3 on page 3: "The home maintains the master copy of the object, from which all cached copies are derived." Nonetheless, Factor does not expressly disclose: *restricting read requests of each and every said computer such that all read requests of each and every said computer are satisfied by reading only the corresponding independent local memory of the requesting computer and not reading from the memory of any other computer.* However, Strom teaches a system of synchronous distributed groupware applications using a replicated architecture (see page 194, last paragraph in the left column). Strom teaches sending notifications to replicated sites when an object is updated (see Figure 3 on page 198 as well as the accompanying description at the bottom right of page 198 to the top left of page 199. Furthermore, see the discussion of propagation at the top right of page 199. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Strom's propagation in Factor's distributed memory in order to provide concurrency control in a distributed memory system as suggested by Strom.

In regard to claim 8, the above rejection of claim 7 is incorporated. Factor further discloses: *wherein the step of modifying the application program is different for different computers.* See page 3, bottom of right column, e.g. "ships the thread to a node chosen by the load balancing function."

In regard to claim 24, the above rejection of claim 7 is incorporated. Factor further discloses: *wherein said program written to operate on only a single computer is a program written to execute within a local processor or processors and local memory coupled to the processor or processors within the single computer.* See at least page 1 right column, e.g.: "It is able to execute a pre-existing, multithreaded application on any given heterogeneous set of machines." Note that the system described in Factor utilizes the Java programming environment which is understood to execute first and foremost on a single processing unit with a single independent local memory. Factor's disclosure implies this by suggesting that their system can take such an application and execute it in a multiple computer system.

13. Claim 9 is rejected under 35 U.S.C. 103(a) as being unpatentable over Factor and Strom as applied in the above rejections of claims 7 and 8, and further in view of U.S. Patent No. 5,802,585 to Scales et al. (hereinafter "Scales").

In regard to claim 9, the above rejection of claim 7 or 8 is incorporated. Factor further discloses:

Factor does not expressly disclose the remaining limitations. However, Scales teaches:

(i) detecting instructions in the unmodified application program which reference the same common memory records; See Scales column 4:38-42, e.g. "coherency."

(ii) listing all such commonly referenced memory records and providing a naming tag for each said listed commonly referenced memory record, See column 6:20-21, e.g. "table," and column 11:6-10, e.g. "ID."

(iii) detecting those instructions which write to, or manipulate the contents of, any of said listed commonly referenced memory records, and See column 4:30-32, e.g. "stores."

(iv) generating an alert instruction following each said detected commonly referenced memory record write or manipulate instruction, said alert instruction forwarding the re-written or manipulated contents and name tag of each said re-written or manipulated listed memory record. See column 1:43-49, e.g. "message passing" and at least column 1 lines 50-57, e.g. "miss check."

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Scales' shared memory with Factor's loading in order to provide coherency (see Scales column 1:20-26)

14. Claims 15-18, 26, 31-32, 34-37, and 39-44 are rejected under 35 U.S.C. 103(a) as being unpatentable over Factor in view of Strom and Scales, and further in view of Strom and U.S. Patent 6574674 to May et al. ("May").

In regard to claim 31, Factor discloses:

A method of compiling or modifying an application program written to include a plurality of instruction code threads intended to execute on and reference only a single computer having a single central processing unit (CPU) or symmetric multiple processing units and a single independent local memory that is not shared with any other computer of a plurality of single computers but said application program to run simultaneously on each one of said plurality of single computers interconnectable via a communications link, with different portions of said application program being simultaneously executable on different ones of said plurality of single computers with each one of the plurality of single computers having the independent local memory accessible only by the corresponding portion of the application program, See at least page 1 right column, e.g.: "It is able to execute a pre-existing, multithreaded application on any given heterogeneous set of machines." Note that the system described in Factor utilizes the Java programming environment which is understood to execute first and foremost on a single processing unit with a single independent local memory. Factor's disclosure implies this by suggesting that their system can take such an application and execute it in a multiple computer system.

Factor does not expressly disclose the remaining limitations. However, Scales teaches:

(i) detecting instructions in the unmodified application program which reference the same common memory records; See Scales column 4:38-42, e.g. "coherency."

(ii) listing all such commonly referenced memory records and providing a naming tag for each said listed commonly referenced memory record; See column 6:20-21, e.g. "table," and column 11:6-10, e.g. "ID."

(iii) detecting those instructions which write to, or manipulate the contents of, any of said listed commonly referenced memory records; and See column 4:30-32, e.g. "stores."

Factor and Scales do not expressly disclose:

(iv) generating and inserting an alert instruction into the unmodified application program to create the modified application program for handling by a distributed run time (DRT) following each said detected commonly referenced memory record write or manipulate instruction indicating that the contents or value of the commonly referenced memory record were re-written or manipulated and may have changed during execution of a code thread, said alert instruction being operative for initiating propagation of the rewritten or manipulated contents and name tag of each said re-written or manipulated listed commonly referenced memory record via the communications link to the

distributed run times (DRTs) of each other of the single computers. However, May teaches that an OM system (i.e. "DRT") is used to handle messages which propagate an indication of write of manipulate instructions of commonly referenced memory records. See column 14 line 65 - column 15 line 23, e.g.

> Each operator message specifies an action and specifies the object on which the action is to be performed. The operator messages include a clear message, a move message, an add message, a replace message, an update message, and a delete message.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Factor's distributed system with May's OM system in order to process operations of shared data as suggested by May.

Factor discloses providing cached copies of an object for further execution on each node. See section 3 on page 3: "The home maintains the master copy of the object, from which all cached copies are derived." Nonetheless, Factor does not expressly disclose: *restricting read requests of each and every said computer such that all read requests of each and every said computer are satisfied by reading only the corresponding independent local memory of the requesting computer and not reading from the memory of any other computer.* However, Strom teaches a system of synchronous distributed groupware applications using a replicated architecture (see page 194, last paragraph in the left column). Strom teaches sending notifications to replicated sites when an object is updated (see Figure 3 on page 198 as well as the accompanying description at the bottom right of page 198 to the top left of page 199. Furthermore, see the discussion of propagation at the top right of page 199. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Strom's propagation in Factor's distributed memory in order to provide concurrency control in a distributed memory system as suggested by Strom.

In regard to claims 15-18, the above rejection of claim 31 is incorporated. Factor does not expressly disclose: *carried out: prior to loading the application program onto each said computer*; *during loading of the application program onto each said computer*; *by just-in-time compilation*; or *by re-compilation after loading*. However, Factor teaches modification of source and byte code. See page 1, right column. Factor also discloses just-in-time compilation. See abstract. The modification of source code is accomplished prior to loading. Finally, Factor discloses instrumentation after first loading the original code. See table 1 on page 6. It would have been obvious to one of ordinary skill in the art at the time the invention was made to try the various sequences source modification since Factor has identified each of these situations with respect to the opportunity for optimization.

In regard to claim 26, the above rejection of claim 31 is incorporated. Factor further discloses: *wherein said program written to operate on only a single computer is a program written to execute within a local processor or processors and local memory coupled to the processor or processors within the single computer.*

In regard to claim 32, the above rejection of claim31 is incorporated. Factor further discloses: *(i) directly notify and propagate* ... See section 3, page 3, middle of left column, e.g. "propagated from a writer to its home as a difference." Factor does not expressly disclose: *to all other DRTs executing on each other one of the plurality of single computers of the re-writing or manipulation and possible change of contents or value of the commonly referenced memory record and then resumes processing; and (ii) indirectly notify and propagate by instructing another thread to notify and propagate the all other DRTs executing on each other one of the plurality of single computers of the re-writing or manipulation and possible change of contents or value of the commonly referenced memory record and then resumes processing.* However, May teaches that notification of manipulation of memory is propagated to DRTs. See column 14 line 65 - column 15 line 23, as cited above. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Factor's notification with May's DRTs in order to efficiently manage data as suggested by May (see column 2 lines 16-35).

In regard to claim 34, the above rejection of claim 31 is incorporated. Factor further discloses: *wherein the communication link comprises the Internet.* See section 2, bottom of page 2, right column.

In regard to claim 35, the above rejection of claim 31 is incorporated. Factor further discloses: *wherein the communication link comprises an intranet.* See section 2, bottom of page 2, right column.

In regard to claim 36, the above rejection of claim 31 is incorporated. Factor further discloses: *wherein the communication link comprises a local area network.* See section 6 on page 6, bottom of left column to the top of the right column.

In regard to claim 37, the above rejection of claim 31 is incorporated. Factor further discloses: *wherein the commonly referenced memory locations comprise JAVA programming language fields and the contents or values stored in the commonly referenced memory locations comprise Java programming field contents or values.* See at least page 1 bottom of right column, e.g. "Java" and "object field."

In regard to claim 39, the above rejection of claim 31 is incorporated. Factor discloses: *wherein application program is written in the JAVA programming language and the step of detecting instructions in the unmodified application program which reference the same common memory records comprise searching through the JAVA programming language code and identifying a put static (putstatic) instruction and generating and inserting an alert instruction into the JAVA application program for each said putstatic instruction so identified.* See section 3 on page 3, middle of left column, e.g. "updates to an object are detected and propagated."

In regard to claim 40, the above rejection of claim 39 is incorporated. Factor further discloses: *modifying the JAVA application program so that during execution of the modified JAVA application program upon executing the inserted alert instruction* *notification, sending the commonly referenced memory record that was re-written or manipulated and may have changed during execution of a code thread with its name tag across the network and receiving the commonly referenced memory record that was re-written or manipulated and may have changed during execution of a code thread with its name tag by a different computer.* See section 3 on page 3, middle of left column, e.g. "updates to an object are detected and propagated."

In regard to claim 41, the above rejection of claim 31 is incorporated. Factor further discloses: *wherein <network communication> is used for a distributed run time (DRT) communication of the commonly referenced memory record that was re-written or manipulated and may have changed during execution of a code thread with its name tag.* See section 2, page 2, bottom of right column, e.g. "Java socket interface."

In regard to claim 42, the above rejection of claim 31 is incorporated. Factor further discloses: *wherein the updating of all of the commonly referenced memory records that were re-written or manipulated and may have changed during execution of code threads are updated over the Internet.* See section 2, page 2, bottom of right column, e.g. "Internet."

In regard to claim 43, the above rejection of claim 31 is incorporated. Factor further discloses: *wherein the communication link comprises the Internet and all updates to commonly referenced memory locations are performed using Internet network packets* *through separate distributed runtimes (DRTs) executing on each of the plurality of single computers.* See section 2, page 2, bottom of right column, e.g. "Internet." Further see May for a teaching of separate DRTs as noted above.

In regard to claim 44, the above rejection of claim 31 is incorporated. Factor further discloses: *writing the value from the <network> for the commonly referenced memory record that was re-written or manipulated and may have changed into the memory location of the receiving computer.* See section 3 on page 3, middle of left column, e.g. "updates to an object are detected and propagated."

In regard to claim 48, the above rejection of claim 7 is incorporated. All further limitations have been addressed in the above rejection of claim 31.

In regard to claim 50, the rejection of claim 45, found below, is incorporated. All further limitations have been addressed in the above rejection of claim 31.

15. Claim 33 is rejected under 35 U.S.C. 103(a) as being unpatentable over Factor, Strom, Scales, and May, and further in view of U.S. Patent 6,101,527 to Lejeune et al. ("Lejeune").

In regard to claim 33, the above rejection of claim 32 is incorporated. Factor, Scales, and May does not expressly disclose: *wherein when the notification and propagation are indirect, the processing of the [code thread that alerts the DRT] is only* *interrupted momentarily before the [code thread that alerts the DRT ~~processing~~ resumes <u>processing]</u> and said another thread which has been notified of the re-written or manipulated commonly referenced memory record then communicates that re-written or manipulated commonly referenced memory record to each of the other single computers so that better utilization of the processing power of various executing threads and gives better scaling with increasing number of single computers when the application program is executed.* However, Lejeune teaches the well-known principle that delegation of tasks to other systems provides greater computing capacity. See column 1 lines 38-43. That is, delegation leads to more computing ability of the system from which the task is delegated. This naturally leads to better utilization of processing power including better scaling. In fact, this concept is the basis of distributed computing. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Factor's distributed computing with Lejeune's task delegation in order to utilize greater computing capacity as suggested by Lejuene.

16. Claims 25 and 29-30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Factor and Strom, and further in view of U.S. Patent No. 6,862,608 to Buhlman et al. (hereinafter " Buhlman").

In regard to claim 25, the above rejection of claim 7 is incorporated. Factor does not expressly disclose: *wherein each of the computers operates with the same application program and data and thus all of the plurality of computers have the same application* *program and data*. However, Buhlman teaches that programs can be operated using the same program and data. See column 1 lines 30-38. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Buhlman's teaching of multiple copies of shared memory with Factor's distributed execution in order to reduce latency as suggested by Buhlman.

In regard to claims 29 and 30, the above rejection of claims 7 and 25 are respectively incorporated. Factor does not expressly disclose: *eliminate clock cycle delays that would otherwise be associated with one or said plurality of computers reading memory physically located in a different one or ones of the plurality of computers formed in a distributed shared memory arrangement*. However, Buhlman discloses this by way of providing updated data values to each of the distributed processors. Each processor has its own copy of data and therefore does not have to request and wait for a data item from another processor. See column 3 lines 48-62. Thus delays associated with waiting for data from another processor are eliminated. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Buhlman's teaching of multiple copies of shared memory with Factor's distributed execution in order to reduce latency as suggested by Buhlman.

17. Claim 38 is rejected under 35 U.S.C. 103(a) as being unpatentable over Factor, Strom, Scales, and May, and further in view of U.S. Patent Application Publication 20030028364 by Chan et al. ("Chan").

In regard to claim 38, the above rejection of claim 31 is incorporated. Factor further discloses: *wherein the commonly referenced memory records comprise JAVA programming language fields*. See at least page 1 bottom of right column, e.g. "Java" and "object field." Factor does not expressly disclose: *the JAVA programming language fields are listed by object and class*. However, May teaches a list of objects. See column 6 lines 23-26. Further, Chan discloses a list of fields in an object class. See paragraph [0003]. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Factor's fields with May and Chan's teaching of field lists in order to easily navigate elements as suggested by Chan.

18. Claims 45-46 are rejected under 35 U.S.C. 103(a) as being unpatentable over May in view of Factor and Strom.

In regard to claim 45, May discloses:

In a multiple computer system including a plurality of single computers interconnectable via an Internet or intranet network communications link, a method of loading an original application program onto each of said plurality of single computers, the original application program having original application program code including a plurality of original code threads all written to execute on and reference a single computer having a single processing unit or symmetric multiple processing units and a single local memory with a local memory capacity that is not shared with any other *single computer of said plurality of single computers, the system configured to enable simultaneous cooperative execution of said application program by said plurality of single computers, with the original application program being modified to form at least one modified application program with different portions of said modified application program being simultaneously executed within a different independent local processor and a different independent local memory within each different one of the plurality of single computers, said different independent local memory within each said different single computer not forming a distributed shared memory arrangement and being accessible during execution of said application program and said different portions of said application program only by the different portion of the application program actually executing within the different local processing unit or symmetric multiple processing units of the different computer,* See May, column 2 lines 16-28 which generally describes application sharing where distributed computers execute the same program and share the same data. Also see Fig. 1, as well as Fig. 2 which shows an exemplary local processor coupled to local memory.

the method comprising the steps of:

loading the application program onto each different computer of said plurality of single computers, said application program including a reference to a program memory field that may be references by one or more of said plurality of computers during execution of their respective different portion of the application program; and See column 2 lines 32-33, e.g. "manages the adding, deleting, and modifying of the shared data."

...

...an insertion of at least one code thread prior to execution that upon execution by one of said single computers initiates a sequence of events that result in a network packet communication over said Internet or intranet network communications link that contains an identifier of the referenced memory field and the contents or value of that memory field. See Fig. 1 elements 104, 114, and 124 which provides at least one code thread which manages the adding, deleting and modifying of the shared data. See column 2 lines 33-35, e.g. "The OM system also controls the transmitting of modifications to the copy of the shared data to the other computers."

May does not expressly disclose:

modifying the application program on each said different single computer before execution of said different portion of the application program on each said different single computer; and

restricting read requests of each and every said computer such that all read requests of each and every said computer are satisfied by reading only the corresponding independent local memory of the requesting computer and not reading from the memory of any other computer;

However, Factor discloses modification of bytecodes to enable distributed execution. See section 4 on page 3, e.g. "Bytecode instrumentation." It would have been obvious to one of ordinary skill in the art at the time the invention was made to use May's application programs with Factor's instrumentation in order to enable an application to be distributed as suggested by Factor.

Factor discloses providing cached copies of an object for further execution on each node. See section 3 on page 3: "The home maintains the master copy of the object, from which all cached copies are derived." Nonetheless, May and Factor do not expressly disclose: *restricting read requests of each and every said computer such that all read requests of each and every said computer are satisfied by reading only the corresponding independent local memory of the requesting computer and not reading from the memory of any other computer.* However, Strom teaches a system of synchronous distributed groupware applications using a replicated architecture (see page 194, last paragraph in the left column). Strom teaches sending notifications to replicated sites when an object is updated (see Figure 3 on page 198 as well as the accompanying description at the bottom right of page 198 to the top left of page 199. Furthermore, see the discussion of propagation at the top right of page 199. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Strom's propagation in Factor's distributed memory in order to provide concurrency control in a distributed memory system as suggested by Strom.

In regard to claim 46, the above rejection of claim 45 is incorporated. May further discloses: *executing said modified application program and generating and communicating said network packet communication over said Internet or intranet network communications link that contains said identifier of the referenced memory field and the contents or value of said referenced memory field.* See column 10 lines 18-23, e.g. "local object handle."

Allowable Subject Matter

19. The following is a statement of reasons for the indication of allowable subject matter:

The examiner indicated that this application would be in condition for allowance if the independent claims 7, 31, and 45 are amended to include the features as indicated in during the 8/26/2009 examiner initiated interview. The discussed features, taken in combination with all remaining features of the independent claim are not taught or suggested by the prior art of record. Applicant has submitted new claims 49 and 51 which provide such allowable limitations, but are dependent upon rejected claims, and have issues related to 35 U.S.C. § 112, second paragraph as indicated above.

20. Claims 52-53 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

Conclusion

21. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

22. Any inquiry concerning this communication or earlier communications from the examiner should be directed to JAMES RUTTEN whose telephone number is (571)272-3703. The examiner can normally be reached on M-F 10:00-6:30.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to JAMES RUTTEN whose telephone number is (571)272-3703. The examiner can normally be reached on M-F 10:00-6:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571)272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/J. Derek Rutten/
Primary Examiner, Art Unit 2192